Computers & Graphics 33 (2009) 607–615

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Knowledge Assisted Visualization

A high-dimensional feature clustering approach to support knowledge-assisted visualization

Julia EunJu Nam *, Mauricio Maurer, Klaus Mueller

Center for Visual Computing, Computer Science Department, Stony Brook University, Stony Brook, NY 11794-4400, USA

ARTICLE INFO

Article history:
Received 16 April 2009
Received in revised form
13 June 2009
Accepted 26 June 2009

Keywords:
Feature descriptors
Classification
Categorization
Data indexing

ABSTRACT

The ever-growing arsenal of methods and parameters available for data visualization can be daunting to the casual user and even to domain experts. Furthermore, comprehensive expertise is often not available in a centralized venue, but distributed over sub-communities. As a means to overcome this inherent problem, efforts have begun to store visualization expertise directly with the visualization method and possibly the dataset, to then be utilized for user guidance in the data visualization, suggesting to the user both the visualization method and its best parameters for the data and task at hand. While this is certainly an immensely useful and promising development, one requirement remains – the matching of a newly acquired dataset with the appropriate segment of the library storing the expert knowledge. This requires one to detect and recognize the dataset's category at some level of granularity and then use this information as a library index. We describe a possible framework for accomplishing the first stage of this process, namely the data categorization, using data classification via a rich set of feature vectors sufficiently sensitive to detect critical variations. We demonstrate the utility of our framework by ways of a set of medical and computational datasets and visualize the resulting categorization as a layout in 2D.

1. Introduction

Knowledge-assisted visualization (KAV) [5] seeks to augment visualization methods as well as data with expert knowledge, in order to make navigation through visualization parameters spaces easier for users unfamiliar with a given method or data visualization in general. Major challenges here are how this knowledge is collected, selected and stored and how it is indexed by data and task. It is obviously infeasible to have an expert preview and navigate each and every newly acquired dataset before it is released to the general public. This may be possible for an isolated set of showcase datasets, but it is impossible in the general case. Hence, we need to somehow learn and then generalize the associations of data with the available collection of expert knowledge. Such a task falls into the general domain of *categorization* problems. It requires a set of rich feature vectors capable of classifying data instances at suitable granularity. In the following, we first motivate the need for rich feature vectors, from a visualization standpoint. The remaining paper is then dedicated to describe a set of feature vectors that we found to work well for a number of diverse data examples. We envision that these feature vectors could ultimately be used as a key to retrieve from a database of expert knowledge the segment of expertise applicable to the dataset at hand. Further augmentation of this key by a specific task as well as user expert level and personal preferences would then be relatively straightforward.

The ultimate purpose of visualization is to exploit the analytical powers of the human brain to integrate low-level features, made visible in the graphics rendering, into a full diagnostic evaluation. While the feature detection is a pre-attentive process, the subsequent integration is a conjunctive one that takes time and conscious attention. In fact, this bottom-up approach constitutes the fundamental underpinnings of *Integration Theory* [32] (as opposed to *Gestalt Theory*, which is top-down), and has also given the new field of *Visual Analytics* its great appeal. Discrete (sampled) data rendering has produced a great variety of techniques to make data features visually salient for easy detection by the human visual system. One measure of distinguishing these many techniques is by the underlying *feature model*, whose parameters are captured in a *feature vector*. Here, the simplest model is the one that is purely sample-based, that is, just the sample value (we shall assume scalar densities for this paper, without loss of generality), which can be visually enhanced via suitable RGB mappings in transfer functions. This leaves it completely up to the user to infer structure in the visual integration process. The next step up is to assist the user in making the conjunctive feature integration. Shape is a strong cue, and to infer it, one can calculate gradients from the sample

* Corresponding author. Tel.: +1 631 632 1524.
E-mail address: ejnam@cs.sunysb.edu (J.E. Nam).

0097-8493/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2009.06.006

neighborhood. More advanced local computations can emphasize higher-level shape features, such as curvature, suggestive contours, and others. All of these features have in common that they have direct visual manifestations which can be readily brought out by their interaction with light. Thus, the underlying models have direct correspondences to familiar real-life expressive artifacts. Similar is true for vector and flow fields where streamlines or deforming textures can be used to visually aid in feature integration. But feature models can also reflect more abstract events, which can be deduced from the data. Popular here are physical as well as topological events in iso-contour and flow fields, which all can be visualized as a feature space of graph-like representations [24,35]. Glyphs have also been devised to visually communicate the presence, structure, and location of such features to the user [34].

In all of the above cases, image analysis, guided by a suitable feature model, was used to perform some low-level feature integration for the user, so these patterns could be perceived, visually encoded by appropriate graphical means. But there are many situations where feature models are not known (yet), or at least have not been formulated by a descriptive analytical process. In this case, one must resort back to raw patterns in the data and derive a model from scratch, using a learning approach (often called *model learning*). Learning requires a specification of features in the data that are characteristic for the sought model, able to distinguish positive from negative candidate instances. This specification is a key to finding an accurate model. Ideally, this process is fully automatic. We accomplish this by first deriving a set of rich feature vectors from the data and then using cluster analysis to learn the feature vectors relevant to the model. Here we will be looking for features that are less intuitive on first sight, and are thus not pre-attentive, requiring computer analysis for their derivation. Then, once these rich feature vectors have been learned, they can be used for subsequent classification, selection, and indexing. In that regard, unlike other works in volume graphics based on machine learning [33] our focus is not segmentation, but model-based object-level classification, detection, and categorization.

Our paper is structured as follows. First, Section 2 discusses relevant work. Then, Section 3 presents the low-level feature detection and description mechanisms we use, while Section 4 describes the high-level learning framework to support classification and mining, and the visualization of the outcomes. Finally, Section 5 presents results of some case studies we conducted with our system, and Section 6 ends with conclusions.

2. Previous work

Voxel clustering has been a frequent mechanism for feature characterization. It was popularized in the pioneering work of Kindlmann and Durkin [14] who identified region (object) boundaries by their characteristic arc-footprint in a 2D density-gradient magnitude scatter plot. However, once there are a sufficiently large number of boundaries, this scatter plot becomes excessively cluttered, making the detection of isolated arcs virtually impossible. For this reason, Kniss et al. [16] devised an innovative dual–domain interaction interface, which enables users to explore local regions of the volume in the volume domain and simultaneously observe the emergence of arcs corresponding to local iso-surfaces within the clutter of the scatter plot domain. An iterative process conducted in this manner then identifies, labels, and paints all volume features. Although the emergence of GPUs enables such real-time interaction, this type of feature detection still requires a significant amount of user skill and time. Further, the approach also pointed to the limits of how many dimensions can be simultaneously managed within such an interactive interface.

Other clustering metrics have also been proposed since then, for example the *LH*-values [30], which encode the two volume values encountered by moving up and down the gradient direction from a given voxel. The *LH* metric is a very informative characterization of a boundary, as it reveals the regions which the boundary separates. The *LH* histogram, however, also operates within a scatter plot of two cluster variables, that is, *L* and *H*, which can result in clutter in the presence of many features. The incorporation of more than two metrics creates what is often referred to as *voxel signatures* [31]. Traditionally these have included densities, first and second derivative magnitudes, but also more advanced statistical measures, such as skew and kurtosis, as well as curvature [15], but one could easily add further metrics, such as the *LH*-values discussed above. The visualization of these high dimensional points is challenging, and most typically an array of 2D projection scatter plots is used. This, however, limits a holistic assessment of the present patterns – the user must reconstruct the possibly complex high-dimensional shape mentally. Dimension reduction methods, such as SOM (Self Organizing Maps) [27], can be used to compress the high-dimensional space into 2D, however, this can be prone to errors and distortions, and it also requires a discretization into bins for display. An early but powerful direct high-dimensional exploration paradigm providing a combined scatter plot of an arbitrary number of variables is the Grand Tour [1]. The user is guided through hyper-space along a trajectory that is decided along the way by selecting the maximum of a given projection pursuit metric. The user stops when the current viewpoint is the local maximum of this metric. In contrast, the method of parallel coordinates [13] maps a high-dimensional point into a piece-wise linear line that spans the vertical dimension axes. With Parallel Coordinates, clusters can be isolated by brushing the appropriate data axes. Another form of displaying high-dimensional data is the spectral plot, as has been used in [23], where various cues were added to help users perceive *N*-dimensional relationships. However, once the number of dimensions becomes even modestly large (dozens to hundreds) all of these paradigms fall victim to the curse of dimensionality [2], which makes the search for interesting patterns an unwieldy task.

To circumvent the scalability problems associated with visual clustering and pattern detection, we choose to instead use more descriptive, yet low-level, feature descriptors, which allow for more informed (pre-integrated feature) clustering in the computational domain, to yield the desired categorization descriptor. A subsequent visualization of the inter- and intra-cluster relationships, along with iconic representation of the underlying patterns, then optionally allows users to get a feel for the semantics of their data. This enables an information-assisted visualization interface [5], in which users, possibly experts, would select visualization method and parameters based on this explanatory data property illustration.

One of the feature descriptors we employ are local density histograms, which have found frequent use in volume rendering before. For example, [22] utilize histograms of local neighborhoods to capture tissue characteristics for locally-sensitive transfer function specification, while [28] employ local histograms to simulate color bleeding and ambient occlusion effects. Texture descriptors have also recently been used by Caban and Rheingans [3]. But apart from densities and higher-order statistics derived from them, we find it also helpful to characterize local features by their gradient field statistics, in form of scale-space gradient histograms. For this, we use the SIFT (Scale Invariant Feature Transform) operator [20,21], which has become quite popular in the computer vision literature and content-based

image retrieval, but to the best of our knowledge has rarely, if at all, been used in the field of volume visualization. We note that our descriptors may be augmented with other high-dimensional feature vector descriptors, such as size [6] and moment statistics [26], as well as with standard local descriptors, such as density and derivatives, for potentially even better results.

3. Low-level feature detection and description

As mentioned, we use two types of statistical low-level feature descriptors: density histograms to represent the overall texture statistics and gradient histograms to capture the perceptual effects of these. In particular the latter is targeted to support the feature integrative systems of human vision, which is the target of any visualization system. Both will be insensitive to affine transformations, such as scale, rotation, and translation, to support feature and object characterization in large datasets. We describe both of these feature descriptors in the following.

A global histogram is not descriptive enough as a classifier, and this motivated the use of more feature-oriented histograms. We did not attempt automated segmentation which is still an open research area, to confine the extent of histograms within object boundaries. Instead, we aimed for a representation that has good potential to capture the essence of an object, and yet is automatic to compute. This led to our approach of computing the density signatures in local histograms at a hierarchy of window sizes, to enable the detection of feature density statistics at multiple levels of scales. For this, we normalized the histograms. We also perform optional smoothing at the histogram level (not the image level) to reach independence of the sampling process that generated the data. Rotation invariance is achieved by constructing the histograms within slightly overlapping radial regions. We also experimented with space-filling tilings: hexagons in 2D, rhombic dodecahedrons in 3D, and so on. For reasons of simplicity in the indexing [37], we eventually chose the radial primitives in our work.

To encode the gradient signatures we used the SIFT feature descriptor, as motivated in Section 2. In the following section, we shall describe the SIFT feature descriptor for the traditional 2D (image) case [21]. Section 3.2 will then describe how we make use of it in time-varying 2D and 3D in light of our application.

3.1. The SIFT feature descriptor

The SIFT feature descriptor [20,21] is based on a model of the behavior of cerebral cortex cells in primate vision, which are sensitive to intensity gradients at different levels of scale. Founded in these evolutionary principles, it has been shown to outperform other local descriptors on both textured and structured scenes. The SIFT algorithm consists of two phases: (i) the detection of critical points (the *keypoints*) in scale-space and (ii) the encoding of these into *keypoint descriptors*. As has been formally shown in [19], the Gaussian forms the best scale-space kernel, and it is used to compute the scale-space used by SIFT for key-point detection. A Gaussian kernel is parameterized by the standard deviation σ:

$$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{-(x^2+x^2)/2\sigma^2} \tag{1}$$

The SIFT scale-space is formed by convolving the original image by a cascade of Gaussians, yielding a set of N images with band limit $2^i\sigma$ and down-sampled by a factor of 2^i, $0\le i\le N-1$. Each image thus represents an octave in scale-space. This pyramid makes features at multiple scales accessible for a localized detection of keypoints, and thus makes features scale-invariant. The detection itself is performed by finding local extrema in a Difference-of-Gaussians (DoG) representation on each pyramid level. A DoG $D(x, y, \sigma)$ is computed by

$$D(x,y,\sigma)=(G(x,y,k\sigma)-G(x,y,\sigma))I(x,y) \tag{2}$$

where $I(x, y)$ is the image and $G(x, y, \sigma)$ the Gaussian kernel. SIFT uses a set of DoG within each octave to enable a finer-scaled and more spatially localized search for local extrema. Each scale-space octave is first subdivided in a set of s images, progressively band-limited with $k^j\sigma$ where $k=2^{1/s}$ and $1\le j\le s$. In fact, we require s+3 blurred images so that the final extrema detection covers the complete octave.

Keypoints are then detected by comparing a point's 8 neighbors on the current scale and the 9 neighbors at the scales above and below. This requires 27 comparisons to determine if the point is a local maximum/minimum. Spatial localization of the keypoint can be improved by a using a Taylor series expansion of the local DoG gradients and setting the derivative to zero [21]. An inherent problem with SIFT is that there can be an abundance of keypoints. Thus additional metrics must be employed to define the relevant saliency of the local extrema. One such metric is contrast, defined by the magnitude of the DoG response's local extremum $|D(x, y, \sigma)_{\max}|$. Typically, a value of $T_{DoG}=0.03$ is used in practice (assuming image values normalized in [0, 1]), providing reductions on the order of 10%. A second criterion is to reject all keypoints that are part of an edge. These can be detected by gauging the local curvature using the point's Hessian matrix H. A point is an edge point if the ratio r of the two principal components is above a certain threshold, say 10. This ratio can be efficiently determined by computing the trace and determinant of H and rejecting the point if the following relation does not hold:

$$\frac{Tr(H)^2}{Det(H)}<\frac{(r+1)^2}{r} \tag{3}$$

This test typically rejects an additional 20–30% of the potential keypoints.

The surviving strong keypoints are now encoded into descriptors which attempt to model the keypoint neighborhood and make it sufficiently distinct for recognition tasks. This encoding is by ways of an orientation histogram of local gradients (measured in the scale space image that is closest to the scale of the keypoint), quantized into bins. It is constructed by computing the magnitude and orientation at each image sample point in a region around the keypoint location, weighted by a Gaussian function with σ equal to one half the width of the descriptor window to achieve a certain level of smoothing. The samples are then aggregated into 8-bin orientation histograms describing the neighborhood over a 4×4 matrix of subregions. Fig. 1 shows a 2D example. This histogram is stored into a 128-long feature vector, along with the scale ID and location.

The SIFT feature descriptor can be made orientation independent and therefore rotation invariant. For this, the highest peak in the orientation histogram is detected and this orientation forms the key-points alignment vector (that is, the orientation histogram is rotated by this vector into a normalized orientation). In addition, new keypoints are also created for all orientation histogram directions with local peaks within 85% of the global peak. This adds typically 15% keypoints, but it brings more stability to matching tasks.

3.2. Extending the SIFT feature descriptor to time-varying 2D and 3D imagery

SIFT has traditionally been used only in 2D in the context of images, but recently [4] and [19] have generalized the framework into higher dimensions, for the purpose of determining accurate

feature point correspondences between medical volume datasets for object alignment and for action recognition in video data, respectively. Although both works extended SIFT to be used with 3-dimensional data, the former employs a more sophisticated keypoint localization approach whereas the latter relies on a random procedure for the extraction of interesting points. This latter methodology allows for faster runtimes but fails to capture the essence of the SIFT algorithm, which is the localization of points that contain striking characteristics of an image. Thus, we use a method similar to that of [4] (and described in Section 3.1) to extract keypoints and compute their feature descriptor in our application.

First a difference of Gaussian (DoG) pyramid of datasets is built. A series of s Gaussian blurred volumes is computed from the original dataset with sigma $=\sigma$, $k\sigma$, $k^2\sigma$, $k^3\sigma$,…, $k^s\sigma$ forming an octave (plus the additional two volumes at either end needed to compute s DoGs). In our experiments, we have used $s=3$ to keep storage reasonable. The first volume of the first octave is the original dataset and the first volume for the subsequent octaves is a half-sized version of the last blurred volume of the previous level. The DoG datasets are then computed from its neighboring blurred datasets at scales $k^j\sigma$ and $k^{j+1}\sigma$. Finally, each DoG volume is searched for local maxima or minima. Similar to the SIFT algorithm, every voxel is compared against all its neighboring voxels in the current scale and in the DoG datasets in the scale above and below. In 3-dimensions, this step requires checking 80 voxels (26 neighbor voxels in the current volume and the 2×27 voxels in the corresponding locations in the adjacent volumes in the same octave). Similar to the 2D case, we use the two metrics, contrast and edge, to cull some of the detected keypoints. For the former, we set $T_{DoG}=0.3$ (assuming voxel ranges in [0, 1]). Using the edge metric is justified since in a full-fledged framework we would prefer to use the 1st and 2nd derivative as a separate feature descriptors, obviating the use of edge keypoints for our SIFT-based feature descriptors. We use a 3D Hessian matrix in Eq. (3) and set $r=10$.

The 3D-SIFT feature descriptor is comprised of a $4\times 4\times 4$ matrix of subregions, each $4\times 4\times 4$ voxels large. This results in a cubic voxel region around the keypoint, with each of the 64 subregions summarized by an 8×8 histogram of voxel orientations (see Fig. 2). The natural generalization of the SIFT gradient direction representation to 3-dimensional datasets uses 2 spherical coordinates (θ, φ) and gives rise to 2-dimensional orientation histograms with b bins for each coordinate, spanning (2π, π) radians. Every gradient magnitude is weighted by a Gaussian function centered at the keypoint before being accumulated into the corresponding bin of the subregion histogram. The concatenation of such histograms form the 3D-SIFT feature descriptor.

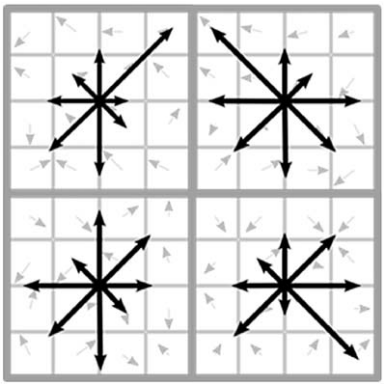

Fig. 1. SIFT keypoint neighborhood. Pre-computed gradients (background) are accumulated into 8-bin orientation histograms (foreground) for each subregion. This figure shows an 8×8 neighborhood with 2×2 subregions, while SIFT typically uses 16×16 neighborhoods with 4×4 subregions.

4. High-level learning and classification, and signature visualization

Having captured the essential perceptual features of a sampled object or phenomenon by way of the feature descriptors described in Section 3, we are now ready to assemble these into high-level constructs. Our goal is to *learn* a high-level representation of the sampled object, using its low-level features. Here, we distinguish between three levels. The level-1 categorization distinguishes different types (categories) of objects to give a general description to each class of objects (for example, smoke vs. fire). The level-2 categorization is run on objects of the same class but distinguishes them by their unique characteristics (for example, turbulent smoke vs. laminar smoke). Finally, the level-3 categorization groups similar individual features within a level-1 or level-2 category. This similarity grouping can be achieved via cluster analysis, such as k-means clustering [9] or affinity clustering [10] and uses the entire feature vector. In k-means clustering the data are partitioned into k clusters, such that each data point belongs to the cluster with the nearest mean. The number of clusters is chosen beforehand, but can be automatically adapted using some quality metric as iterations proceed [25]. Affinity clustering, on the other hand, simultaneously considers all data points as potential center points (called exemplars) and gradually identifies clusters by iterative message passing among points, "electing" the

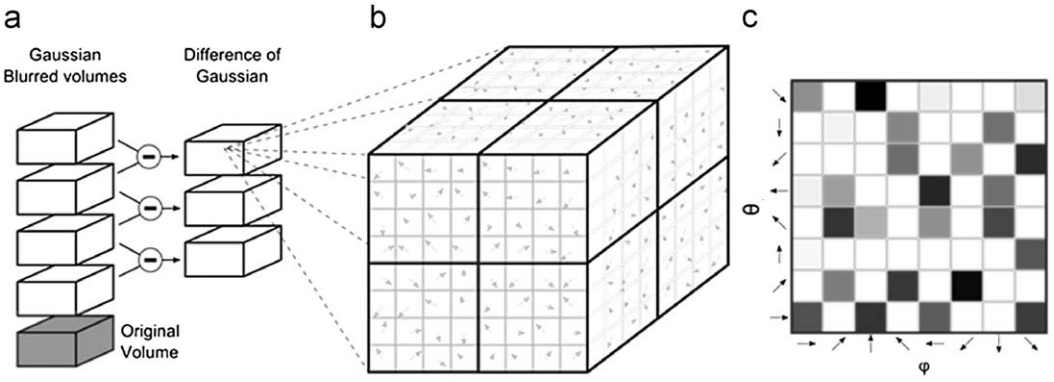

Fig. 2. Summary of the 3D-SIFT algorithm: (a) One octave of the pyramid. The scale increases along the up-direction. (b) Depiction of subregions adjacent to a keypoint. The figure shows only 2×2 subregions while the algorithm employs 4×4 regions. (c) 2-Dimensional histogram of gradient orientations of one subregion. Black squares correspond to 1 and white squares to 0. The histogram is normalized to [0, 1].

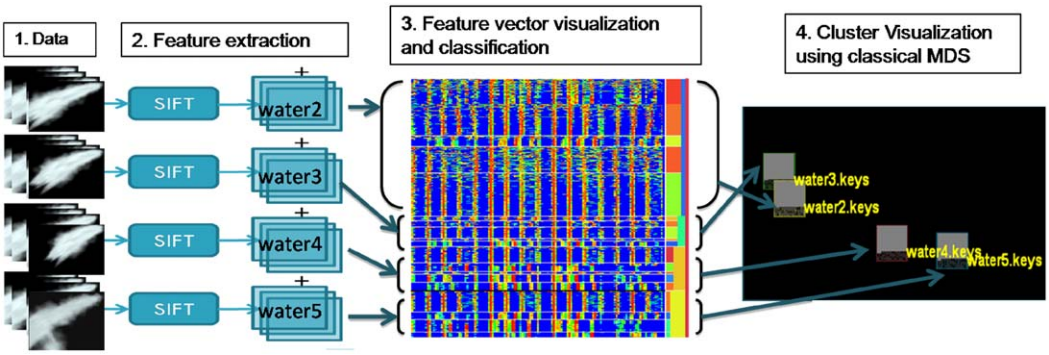

Fig. 3. The main pipeline. The dataset used for illustration includes four different 2D water jet flows (water2 through water5, from top to bottom). All jet flows have the same direction: ejected from the top-right corner to the bottom-left corner. They have different speeds from slowest (water2) to fastest (water5). Water5 flows slightly to the side direction partially: (1) the four instances of the 2D water jet flow dataset; (2) computing the SIFT feature descriptors for each dataset and creating a set of feature vectors; (3) for each dataset, performing *k*-means clustering of the feature vectors into a set of sub-features; and (4) visualizing the clusters in an MDS map to convey their relative similarities.

most agreeable centers. We note that the top two levels may overlap in terms of the third level, that is, third-level features may be part of separate categories and instances thereof.

We generated a number of animated sequences of water flows to experiment with our framework (see Fig. 3 for an illustration of the process). First, we computed SIFT feature descriptors of five different water flows (panel 1). We then used the SIFT feature descriptors of each dataset to form the high-level categorizations, level-1 and level-2, and ran *k*-means clustering for the level-3 categorization. A value $k = 5$ worked best for our experiments, but we could also have used adaptive *k*-means or affinity clustering to gain more independence from choosing an initial *k*. Now, each dataset has five clusters of feature descriptors, and each cluster in each dataset constitutes an individual feature. If the user runs clustering on an individual feature to create sub-features, a two-level hierarchy is formed and can be categorized at the lower level. In fact, the clustering can be performed at any level of the hierarchy using different parameters and various clustering methods. Panel 3 shows a spectral plot [23] of the separated clusters. Each row is one feature vector where value is coded as color (from blue to red, the value is increasing). However, the complexity of this spectral plot prohibits one to discern cluster relationships in an intuitive fashion. We therefore added further support by visualizing the clusters (at all three levels) via Multidimensional Scaling (MDS) [17]. For this, we computed the distances between all cluster pairs using various distance metrics (explained further below) and stored them into an $M \times M$ distance matrix, where M is the number of clusters. There are many methods to embed this distance matrix into a 2-dimensional plane, preserving the original *N*-D distances as close as possible [7]. We chose the classical form of MDS which uses the top two principal components of this distance matrix as the embedding coordinates (see panel 4). The closer the distance the more similar are the clusters, according to the chosen metric. The composition of the MDS map depends on the categorization level of interest. In Fig. 3, each point in the MDS map corresponds to each dataset since we are interested in the level-2 instance categorization. The computed cluster distances, shown in the MDS maps, are an important part of the categorization algorithm, so we shall examine the various methods we used in the following.

We have tried three similarity metrics to calculate the distances for the MDS map: (1) simple Euclidean distance between cluster centers, (2) Euclidean distance between all data points in the cluster, and (3) *CURE similarity*. The CURE similarity is a distance metric which we borrowed from the CURE clustering algorithm [36]. In this scheme, well-scattered representatives are selected in each cluster and their Euclidean distances are

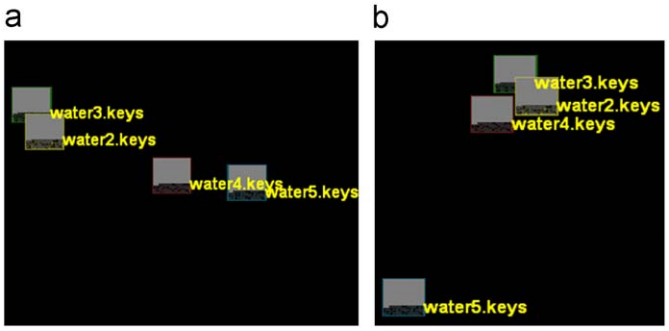

Fig. 4. Cluster visualization using two different similarity measurements for MDS: (a) Euclidean distance of cluster centers; (b) CURE similarity: distances between 15 well-scattered representatives. Only the CURE similarity metric can successfully separate the very different water5 flow.

computed as a metric for similarity. We selected 15 data points to represent each cluster. While the Euclidean distance tends to find neighbors related on the cluster centers, the CURE similarity better represents clusters of arbitrary shape. Fig. 4(a) shows an MDS plot generated by the Euclidian center distance metric. Here, water2 and water3 flows appear very similar, and likewise water4 and water5. However, by visual inspection of the original flows (panel 1), this is not true. Indeed, if the cluster shape is not uniformly distributed about the cluster center, this metric may not capture the similarity correctly in terms of shape. For better results (Fig. 4(b)), we use the CURE similarity metric. Here, water4 is gathered with water2 and water3, while water5 is well separated from the group, which is also evident in the images of these flows. This indicates that while water5 may have a similar cluster center than water2, 3, and 4, it does not have a similar shape within the group.

In order to give the user further insight into the cluster's composition we represent each MDS node as an icon that depicts a matrix of the dimension values. From the bottom left to the top right corner of the each node, the matrix is divided into the number of dimensions. Each cell encodes the average value of each dimension as brightness. This small icon can provide users with insight on the activities in all dimensions in a comprehensive and space-efficient way.

Let us now drill into the MDS plot to assess the cluster decomposition and explore the intra- and inter-cluster relationships. This exposes the level-3 features in relation to the level-2 instances (still using only one level-2 category: water flow). As mentioned above, we run *K*-means clustering on each flow using $K = 5$ (giving rise to the second level of the hierarchy shown on

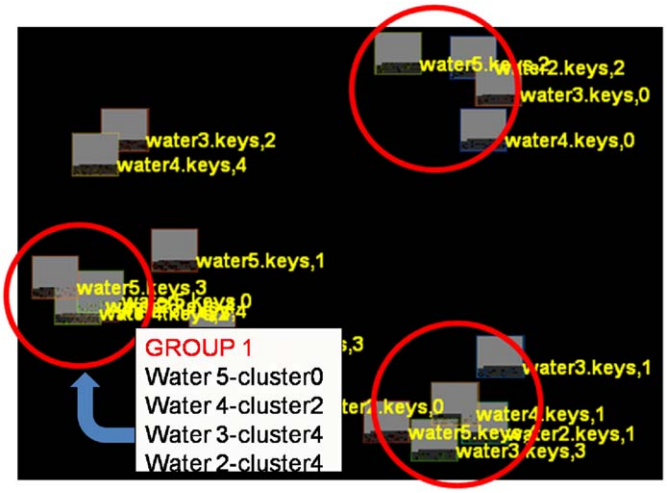

Fig. 5. Separating out individual sub-features (level-3 clusters) in the level-2 clusters using *K*-means clustering. Sub-feature clusters are gathered into three larger groups (red circles) (for interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

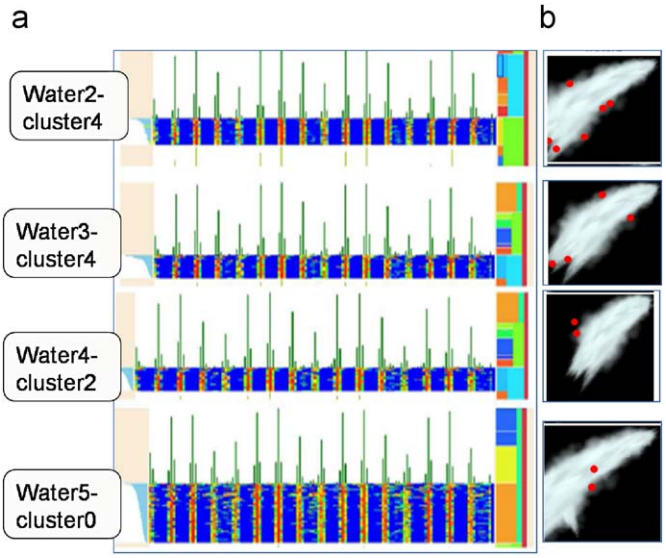

Fig. 6. Clusters in Group 1 in Fig. 5: (a) Clusters represented in spectral plots. The colored patterns are very similar. The histogram (green bars above each spectral plot) also shows similar patterns. (b) Some example positions of keypoint descriptors (red dots) in each corresponding cluster (for interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article).

the right edge of Fig. 4, panel 3). Fig. 5 shows the entire MDS map (for example, water3.keys-2 denotes the 2nd cluster of the water3 flow). We observe that small clusters from different flow datasets overlap at several locations. The spectral plot in Fig. 6 takes a closer look into the feature vectors of the clusters inside the left circle, where we indeed observe that water2–4, water3–4, water4–2, and water5–0 are very similar.

Using these learnt relationships, we could now easily categorize new water flow datasets at all 3 levels. But in a more general context, an important finding from this experiment is that in general there is not a unique set of features that defines a given dataset, but a (random) subset of features might be shared among datasets of the same type, or even of different types or domains. Yet, the combination of these features will likely be unique (which was certainly true for the water flows we used to illustrate our framework). Therefore, as a direct consequence for knowledge-assisted visualization, it will depend on the level of detail (or aspects) one seeks to visualize to determine which knowledgebase segments would need to be indexed (and which would be shared). This was also our motivation for adding the various methods for providing visual feedback, which allow a visual steering of the various parameters. This visual interactive interface will be useful when adding datasets to the knowledgebase which we consider in some sense a knowledge design activity. In the knowledge retrieval stage, all analyses would of course be automated, using the stored parameters, or possibly tunable by a single slider encoding a suitable combination of parameters [29].

5. Results

To evaluate the categorization power of our framework in a more general context, we tested it on a set of medical image data, such as various body parts scanned with MRI and CT, as well as on datasets obtained from fluid dynamics simulations. Further, we also compared the results obtained when treating animated data as a set of independent frames or combined as a time-series. All medical data were obtained from [39].

5.1. Categorization of MRI and CT volumes

We tested four different medical datasets: a knee, a chest, and two head datasets (see side panel of Fig. 7 for an arbitrary rendering of these – the feature descriptors were computed on the raw data). We computed the global and local density histograms as well as the 3D SIFT feature descriptors for each dataset. For the

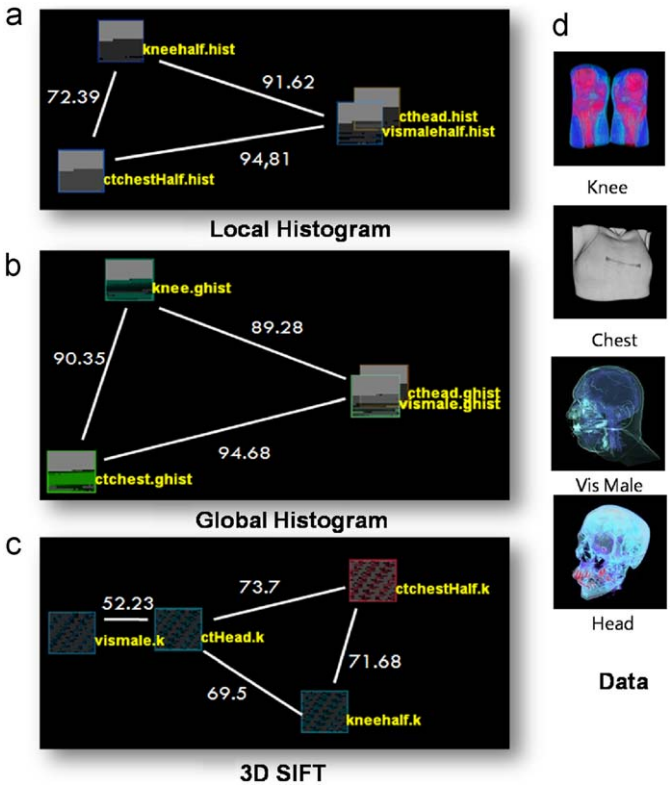

Fig. 7. MDS plot of four CT datasets of different body parts. Categorization via: (a) local histograms, (b) global histograms, and (c) 3D SIFT. We observe that 3D SIFT separates the data objects best.

calculation of the local density histograms, the rhombic dodecahedrons were inscribed in a sphere with radius equal to 8 voxels extended 12.5% to allow for the overlapping of adjacent spheres. At each successive level, the radius of the sphere was doubled, eventually reaching one fourth of the smallest dimension of the volume. The extraction of 3D SIFT feature descriptors was executed with T_{DoG} set equal to 0.0075 (considering the voxels were normalized in the range [0,1]), $\sigma = 4.0$, 3 DoG datasets per octave and 8 bins for the gradient orientation histogram (for each spherical coordinate).

Fig. 7 shows the MDS plots for each feature descriptor. We make the interesting observation that while all feature descriptors are able to categorize the knee, the chest, and the heads, only 3D SIFT is sufficiently sensitive to distinguish the two instances of the head, keeping a smaller feature space distance than to the very different datasets. This distance may then subsequently be used to steer the applied knowledge to the finer distinguishing properties in the KAV process.

Fig. 8 compares the categorization of MRI data with the three feature descriptors. These data consist of three brain scans and a leg scan (sample renderings are shown in the side panel). Here we observe that the global histogram descriptor is only able to distinguish brain data from non-brain (here knee) data. The local histogram descriptor, on the other hand, distinguishes the various brains, but categorizes the knee and one of the brains into overlapping bins, which is less desirable. Finally, the 3D SIFT descriptor spaces the knee and all brains far apart, while detecting a closer match between two of the brains.

5.2. Categorization of flow data

The first set of flow data we tested were time-sequences generated with the program Wondertouch Particle Illusion 3.0 [38]. We used this program since it allowed us to generate a highly diverse set of amorphous phenomena, which we recorded as a sequence of images, one for each frame. Specifically, we generated sets of instances of three (level-1) flow categories: water jet, smoke, and fire. Section 4 already presented the level-2 and level-3 analysis of the water jet category – in this current section we shall now concentrate on the level-1 categorization. We only used SIFT for the flow data since we found it to be superior to the histogram-based descriptor.

We first applied (2D) SIFT to each frame separately. All keypoint descriptors were then concatenated to form one single set of feature descriptors for the entire sequence. In addition, in order to examine if time-coherency can yield better categorization power, we also performed 3D SIFT on the volume assembled by stacking all frames of the flow in order. In such cases, the depth of the volume reflected the number of frames, while the width and height correspond to the width and height of the flow sequence. All parameters were the same as those used with the CT datasets of the previous section.

Fig. 9 presents two MDS plots from the fire category generated with the CURE similarity metric based on both SIFT and 3D SIFT descriptors (Fig. 9(a) and (b), respectively). Although almost imperceptible, the differences between flows 1, 2, and 3 rely on the speed and angle of emission of the particles source. On the other hand, both flows 5 and 6 are unique instances in this category being generated with very different parameter settings, while flow 6 is somewhat related to flows 1, 2, and 3. Being presented with these flow sequences, we notice that SIFT fails to categorize flows 3 correctly. In addition flows 6 and 5 are also (incorrectly) categorized into the cluster of flows 1, 2, and 3. On the other hand, 3D SIFT, taking temporal features and coherencies into account creates a meaningful grouping and categorization.

The next experiment tested our framework on a set of standard time-varying 3D datasets generated in scientific flow simulations: Jet, Vortex, and Shockwave (pictured in that order in Fig. 10(d)). Global and local histograms as well as 3D SIFT feature descriptors were computed and evaluated. Five (3D) frames were extracted

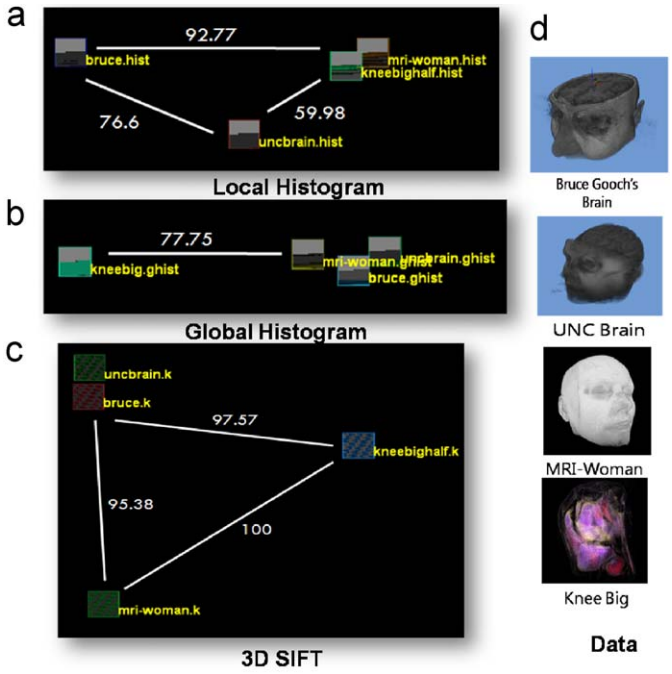

Fig. 8. MDS plot of four MRI datasets of different body parts. Categorization via: (a) local histograms, (b) global histograms, and (c) 3D SIFT; and (d) An example rendering of the data. We observe that the knee scan can be well separated from the three brain scans, which also can be separated.

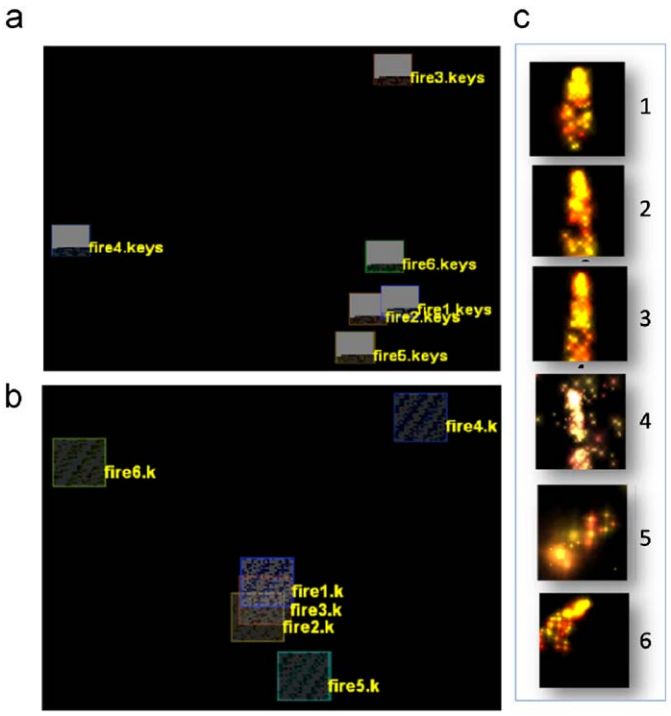

Fig. 9. Comparing the 2D SIFT and 3D SIFT keypoint descriptors for the fire flow category: (a) plot generated with SIFT keypoint descriptors, (b) plot generated with 3D SIFT keypoint descriptors, (c) a selected frame for each the six flow sequences. We observe that the time-coherent (3D) SIFT performs significantly better in the categorization task.

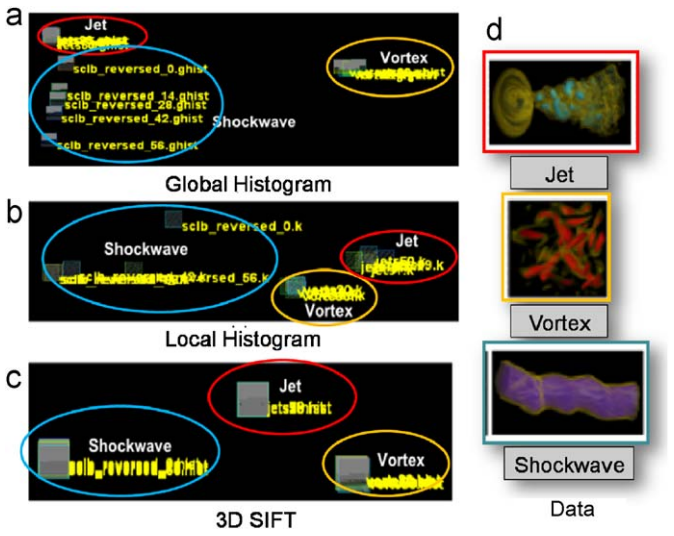

Fig. 10. (a)–(c) MDS plots of three volumetric flow datasets categorized via three different feature descriptors. Clusters are circled in the same color than the frames in the data renderings shown in (d).

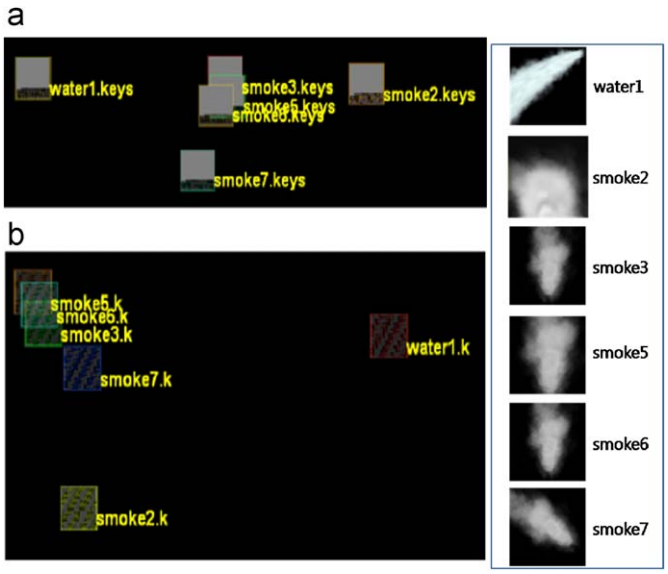

Fig. 11. MDS plots of the smoke category against an unknown flow (water jet flow 1). Plots generated with (a) 2D SIFT, and (b) 3D SIFT keypoint descriptors. We observe that we can successfully differentiate the foreign water flow instance from the smoke category.

from each flow and processed according to the global histogram, local histograms and 3D SIFT algorithms. These frames correspond to the first and last frames and frames located at 25%, 50%, and 75% in the sequence. Fig. 10 illustrates the results we obtained with (a) global histogram, (b) local histogram, and (c) 3D SIFT descriptors (the Shockwave is labeled *sclb_reversed* in these plots). Each node shown corresponds to one frame. Using global histograms, MDS segregates properly both the Vortex and Jet datasets but is not able to correctly cluster frames from the Shockwave dataset. If, on the other hand, local histograms are used as the descriptor, the diagram shows that all frames are correctly clustered except for the first frame of the Shockwave dataset, which is far from the other frames. This is because in the first frame the wave has not started yet and the histogram distribution is rather narrow. 3D SIFT descriptors provide a good clustering as well, but here the first frame is not differently classified. Thus, overall, the local histogram feature descriptor appears to capture the diversity of the individual frames better (the level-2 categorization), while 3D SIFT appears to be more descriptive in the level-1 categorization.

As a final experiment we tested the framework's ability to detect and separate an instance from a foreign category in a set of instances of a known category (which is somewhat related to the method's sensitivity and specificity). Fig. 11 presents MDS plots of our attempt to detect a water jet instance in a set of diverse smoke simulations. We observe in the side panel of Fig. 11 that smoke 3, 5, and 6 are different only in terms of speed and angle of emission. Smoke 7 is a rotated version of the other flows, while smoke 2 is very different from the others in all aspects. We then added a completely different water jet flow to verify the validity of the level-1 and level-2 categorizations. Indeed. both 2D SIFT (a) and 3D SIFT (b) are able to detect the water jet outlier by grouping it further apart than the instances within the smoke category, although 3D SIFT does somewhat better in this.

6. Conclusions

We have described a framework that expresses sampled datasets common in the fields of volume graphics and visualization as a set of rich, yet general, salient low-level feature vectors, and then uses these to create high-level models of the sampled data objects. We aimed for a general framework to support classification tasks needed for indexing the knowledge base in future knowledge-assisted visualization (KAV) systems. We demonstrated these capabilities for a variety of scenes of sampled objects and phenomena, consisting of medical and flow data.

At this stage, we have built the models only by ways of *k*-means clustering to determine descriptive feature groupings based on similarity. This worked already quite well. However, a richer set of object dichotomies could be obtained by introducing probabilistic techniques, such as EM, into the framework, to discover more specific objects in the data [18]. Another useful extension of our framework would be to associate the learned object and feature models with information on appropriate rendering parameters, learned from user studies [12] or automated observers, such as Daly's Visible Differences Predictor (VDP) [8], or a combination of these. Here, it will be probably useful to also include other low-dimensional feature descriptors, such as density, gradients, *LH*, and the like, for more robust categorization results.

Any low-dimensional embedding, including MDS, has residual errors, which result from imperfect mappings of *N*-D to 2D space. Our current work is directed towards using more accurate non-linear MDS-based space embeddings, via force-directed methods [11], in place of our linear PCA-based method. Finally, we also plan to incorporate 4D SIFT descriptors to capture the temporal coherencies in time-varying 3D volumes.

References

[1] Asimov D. The grand tour: a tool for viewing multidimensional data. SIAM Journal on Scientific and Statistical Computing 1985;6(1):128–43.
[2] Bellman A. Adaptive control processes. Princeton: Princeton University Press; 1961.
[3] Caban J, Rheingans P. Texture-based transfer functions for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics 2008;14(6):1364–71.
[4] Cheung W, Hamarneh G. N-SIFT: *N*-dimensional scale invariant feature transform for matching medical images. IEEE International Symposium on Biomedical Image Processing (ISBI) 2007:720–3.

[5] Chen M, Ebert D, Hagen H, Laramee RS, van Liere R, Ma K-L, et al. Data, information and knowledge in visualization. IEEE Computer Graphics and Applications 2009;29(1):12–19.
[6] Correa C, Ma K-L. Size-based transfer functions: a new volume exploration technique. IEEE Transactions on Visualization and Computer Graphics 2008;14(6):1380–7.
[7] Cox T, Cox M. Multidimensional scaling. London: Chapman & Hall; 2001.
[8] Daly S. The visible differences predictor: an algorithm for the assessment of image fidelity. Digital image and human vision. Cambridge, MA: MIT Press; 1993. p. 179–206.
[9] Duda R, Hart P, Stork D. Pattern classification, 2nd ed. New York: John Wiley; 2001.
[10] Frey B, Dueck D. Clustering by passing messages between data points. Science 2007;315:972–6.
[11] Fruchterman T, Reingold E. Graph drawing by force-directed placement. Software: Practice and Experience 1991;21(11):1129–64.
[12] Giesen J, Mueller K, Schuberth E, Wang L, Zolliker P. Conjoint analysis to measure the perceived quality in volume rendering. IEEE Transactions on Visualization and Computer Graphics 2007;13(6):1664–71.
[13] Inselberg A, Dimsdale B. Parallel coordinates: a tool for visualizing multi-dimensional geometry. IEEE Visualization 1990:361–78.
[14] Kindlmann G, Durkin J. Semi-automatic generation of transfer functions for direct volume rendering. ACM/IEEE Volume Visualization Symposium 1998:79–86.
[15] Kindlmann G, Whitaker R, Tasdizen T, Möller T. Curvature-based transfer functions for direct volume rendering: methods and applications. IEEE Visualization 2003:513–20.
[16] Kniss J, Kindlmann G, Hansen C. Multidimensional transfer functions for interactive volume rendering. IEEE Transactions on Visualization and Computer Graphics 2002;8(3):270–85.
[17] Kruskal J, Wish M. Multidimensional scaling. Beverley Hills, CA: Sage; 1977.
[18] Li F, Fergus R, Perona P. A Bayesian approach to unsupervised one-shot learning of object categories. International Conference on Computer Vision (ICCV) 2003:1134–41.
[19] Lindeberg T. Scale-space theory: a basic tool for analyzing structures at different scales. Journal of Applied Statistics 1994;21(2):224–70.
[20] Lowe D. Object recognition from local scale-invariant features. International Conference on Computer Vision (ICCV) 1999:1150–7.
[21] Lowe D. Distinctive image features from scale-invariant key points. International Journal on Computer Vision 2004;60(2):91–110.
[22] Lundström C, Ljung P, Ynnerman A. Local histograms for design of transfer functions in direct volume rendering. IEEE Transactions on Visualization and Computer Graphics 2006;12(6):1570–9.
[23] Nam E, Han Y, Mueller K, Zelenyuk A, Imre D. ClusterSculptor: a visual analytics tool for high-dimensional data. IEEE Symposium on Visual Analytics Science and Technology 2007:75–82.
[24] Pascucci V, Cole-McLaughlin K. Efficient computation of the topology of level set. IEEE Visualization 2002:187–94.
[25] Pappas T. An adaptive clustering algorithm for image segmentation. IEEE Transactions on Signal Processing 1992;40(4):901–14.
[26] Patel D, Haidacher M, Balabanian J, Gröller E. Moment curves. In: Proceedings of the IEEE Pacific Visualization Symposium, 2009.
[27] Pinto F, Freitas C. Design of multi-dimensional transfer functions using dimensional reduction. In: Eurographics/IEEE VGTC symposium visualization (EuroVis), 2007. p. 130–7.
[28] Ropinski T, Meyer-Spradow J, Diepenbrock S, Mensmann J, Hinrichs K. Interactive volume rendering with dynamic ambient occlusion and color bleeding. Computer Graphics Forum 2008;27(2):567–76.
[29] Rezk-Salama C, Keller M, Kohlmann P. High-level user interfaces for transfer function design with semantics. IEEE Transactions on Visualization and Computer Graphics 2006;11(5):1021–8.
[30] Sereda P, Vilanova Bartrolí A, Serlie I, Gerritsen F. Visualization of boundaries in volumetric data sets using *LH* histograms. IEEE Transactions on Visualization and Computer Graphics 2006;12(2):208–18.
[31] Tenginakai S, Lee J, Machiraju R. Salient ISO-surface detection with model-independent statistical signatures. IEEE Visualization 2001:231–8.
[32] Treisman A, Gelade G. A feature-integration theory of attention. Cognitive Psychology 1980;12(1):97–136.
[33] Tzeng F, Lum E, Ma K-L. An intelligent system approach to higher-dimensional classification of volume data. IEEE Transactions on Visualization and Computer Graphics 2005;11(3):273–84.
[34] van Walsum T, Post FH, Silver D, Post FJ. Feature extraction and iconic visualization. IEEE Transactions on Visualization and Computer Graphics 1996;2(2):111–19.
[35] Weinkauf T, Theisel H, Shi K, Hege H-C, Seidel HP. Extracting higher order critical points and topological simplification of 3D vector fields. IEEE Visualization 2005:71–8.
[36] Guha S, Rastogi R, Shim K. CURE: an efficient clustering algorithm for large databases. In: Proceedings of the ACM-SIGMOD international conference on management of data, 1998. p. 73–84.
[37] White D, Jain R. Similarity indexing with the SS-tree. IEEE Data Engineering (ICDE) 1996:516–23.
[38] Wondertouch Particle Illusion 3.0. 〈http://www.wondertouch.com〉.
[39] The Volume Library: 〈http://www9.informatik.uni-erlangen.de/External/vollib〉.